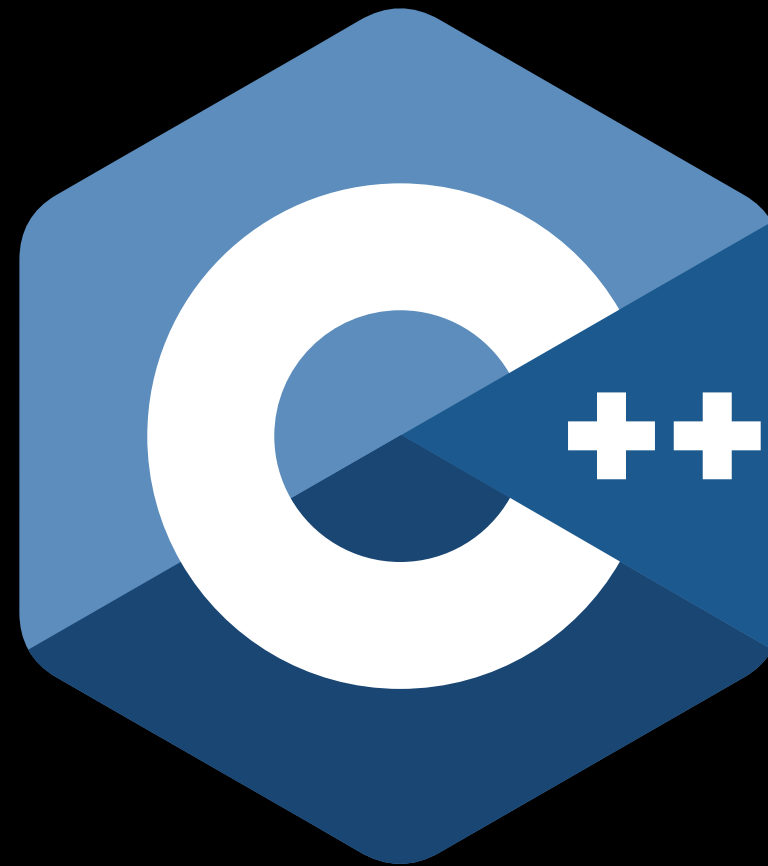


Introducción a C++



C++: Un mejor C

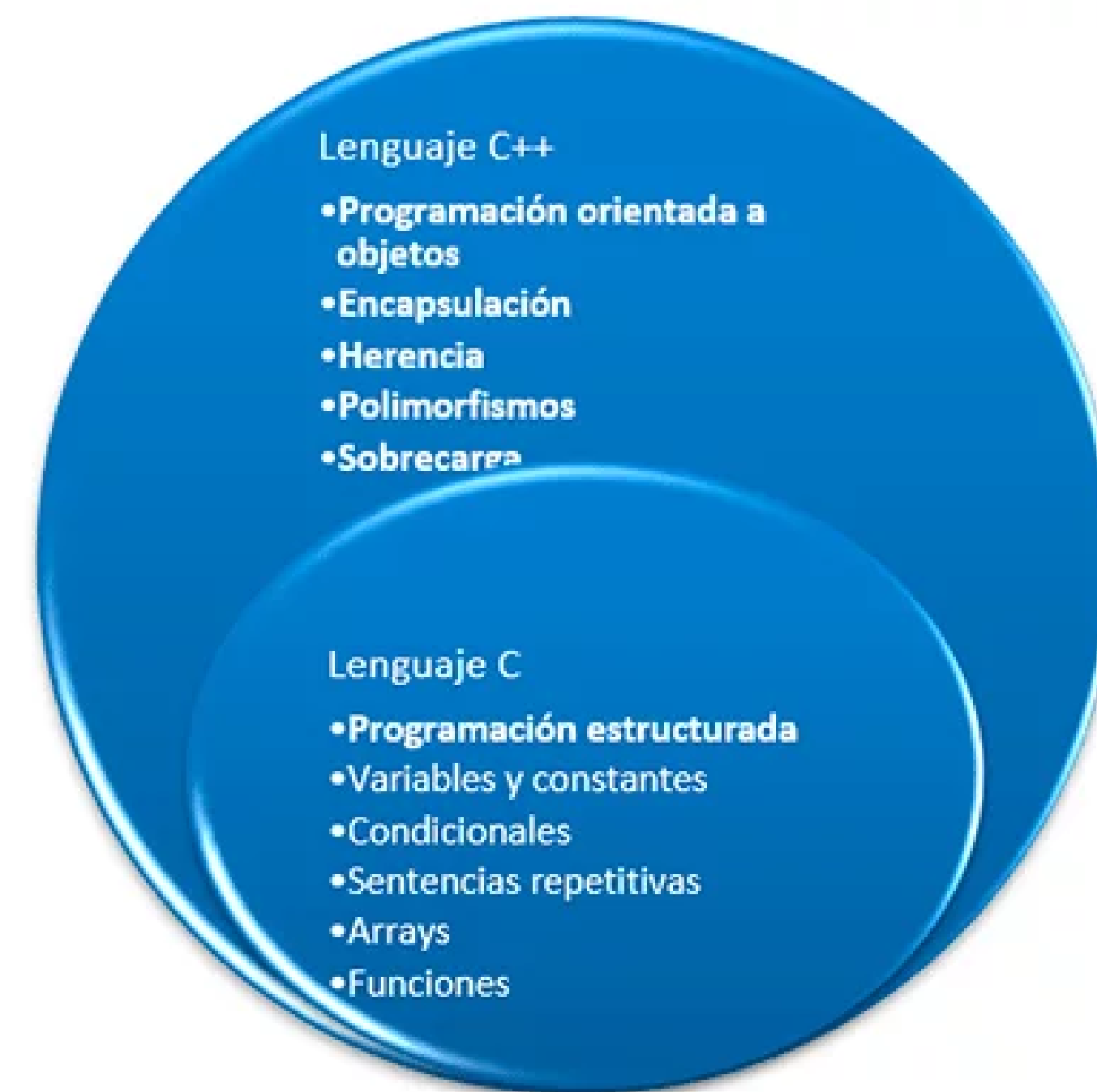
C++: Un mejor C

El lenguaje C

- ✓ Lenguaje creado por Dennis M. Ritchie en 1972
- ✓ Lenguaje de nivel medio:
 - Estructuras típicas de los lenguajes de alto nivel
 - Construcciones para control a nivel de máquina
- ✓ Lenguaje sencillo (pocas palabras reservadas)
- ✓ Lenguaje estructurado (no estrictamente estructurado en bloques)
- ✓ Compartimentalización de código (funciones) y datos (ámbitos)
- ✓ Componente estructural básico: la función (subprograma)
- ✓ Programación modular
- ✓ Distingue entre mayúsculas y minúsculas
- ✓ Palabras reservadas (o clave): en minúsculas

C++ vs C

- C++ es un lenguaje más actual que C
- C++ mantiene todo el poder de C agregando características que facilitan la programación.
- C++ permite la programación orientada a objetos y programación genérica. A diferencia de Java, C++ permite este tipo de programación de forma opcional: Se pueden crear programas en el paradigma procedural u orientado a objetos según se necesite. ¡Incluso mezclas entre ellos!
- C++ posee una biblioteca de funciones mucho más amplia C. Las bibliotecas ya existentes de C son en su mayoría compatibles con C++



Sintaxis de los lenguajes de programación

Los lenguajes de programación

Sintaxis y semántica de los lenguajes

Sintaxis

- Reglas que determinan cómo se pueden construir y secuenciar los elementos del lenguaje

Semántica

- Significado de cada elemento del lenguaje
¿Para qué sirve?

Un primer programa en C++

Bjarne Stroustrup (1983)

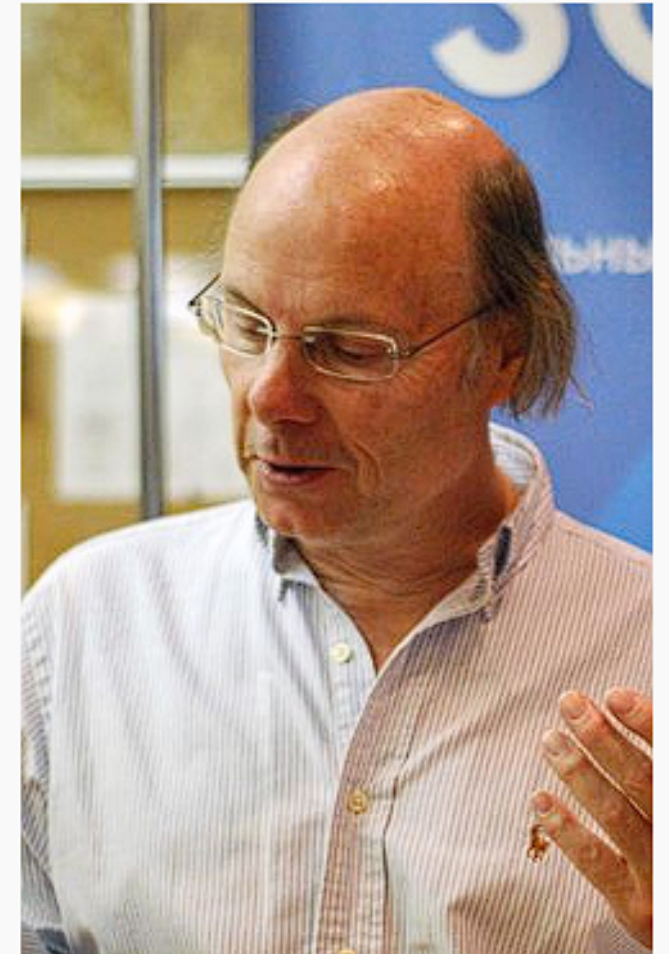
```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hola Mundo!" << endl;
    // Muestra Hola Mundo!

    return 0;
}
```

Hola Mundo!

Bjarne Stroustrup



Bjarne Stroustrup en 2010

Inventé C++, escribiendo las primeras definiciones, y produciendo la primera implementación... elegí y formulé los criterios de diseño para C++, confeccioné también sus principales utilidades, y fui responsable de elaborar extensas proposiciones en el comité de estándares de C++.

Elementos del lenguaje

Instrucciones

Datos: literales, variables, tipos

Subprogramas (funciones)

Comentarios

Directivas

...

```
#include <iostream>
using namespace std;
```

Directiva

Subprograma

```
int main()
```

```
{
```

Dato

Instrucción

```
cout << "Hola Mundo!" << endl;
```

```
// Muestra Hola Mundo!
```

Comentario

Instrucción

```
return 0;
```

Dato

```
}
```

Un primer programa en C++

Hola Mundo!

Un programa que muestra un saludo en la pantalla:

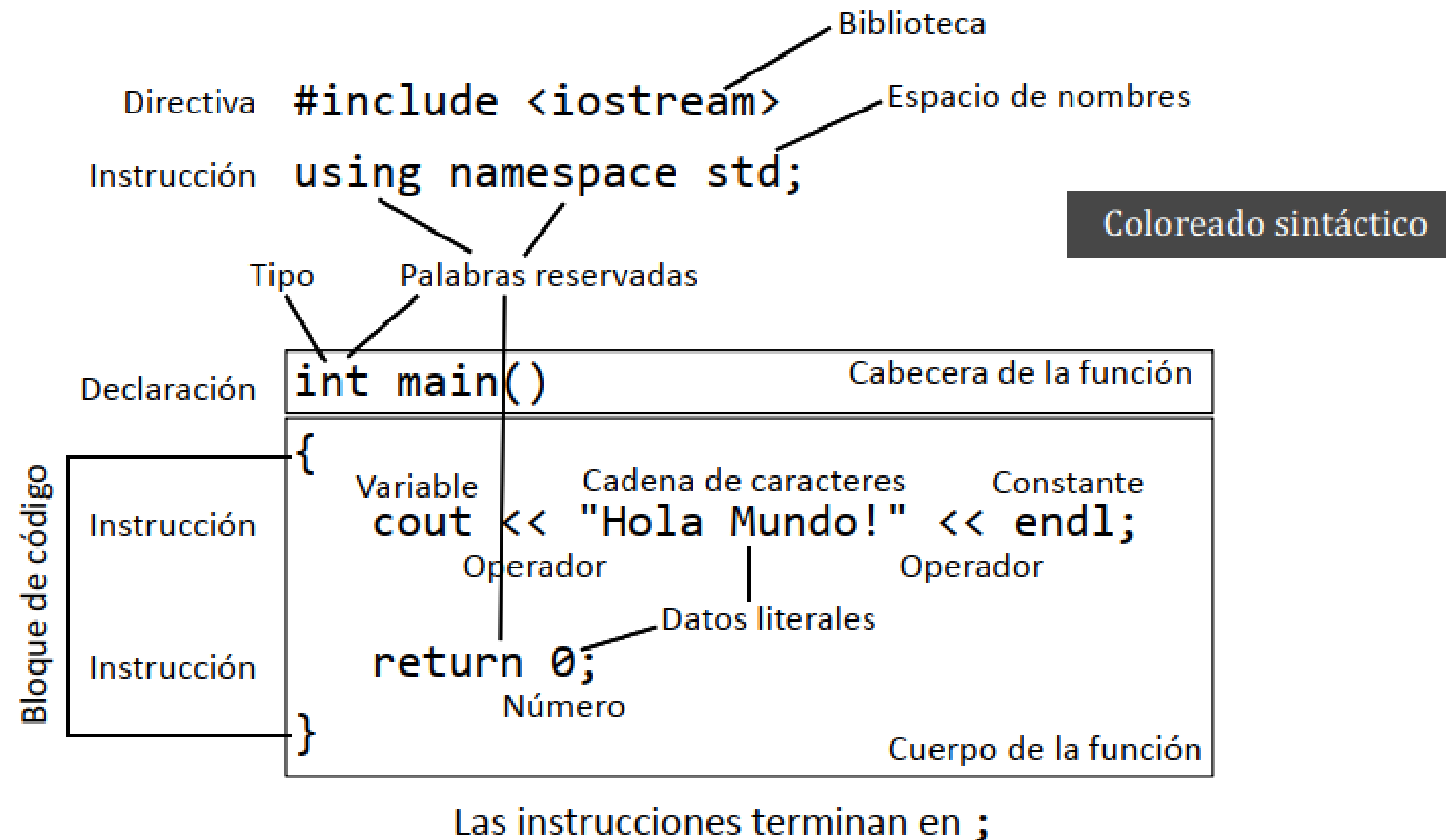
```
#include <iostream>
using namespace std;

int main()
// main() es donde empieza la ejecución
{
    cout << "Hola Mundo!" << endl; // Muestra Hola Mundo!

    return 0;
}
```

Un primer programa en C++

Análisis del programa



Un primer programa en C++

Hola Mundo!

Casi todo es *infraestructura*

Sólo

```
cout << "Hola Mundo!" << endl
```

hace algo palpable

La infraestructura (notación, bibliotecas y otro soporte)

hace nuestro código simple, completo, confiable y eficiente

¡El estilo importa!

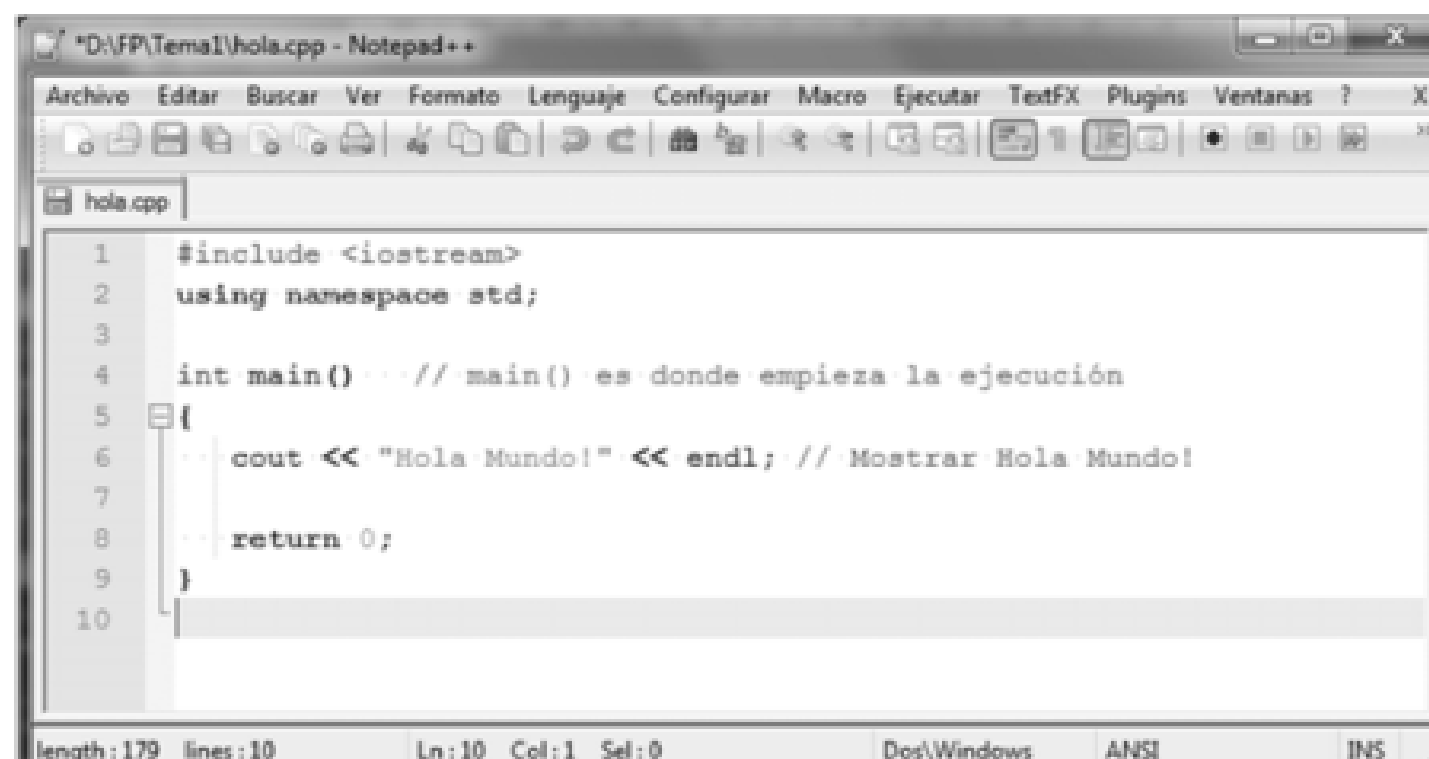
Fundamentos de la programación

Herramientas de desarrollo

Herramientas de desarrollo

Editor

- ✓ Bloc de notas, Wordpad, Word, Writer, Gedit, Kwrite, ...
(texto simple, sin formatos)
- ✓ Editores específicos: coloreado sintáctico
- ✓ Recomendación: Notepad++



```

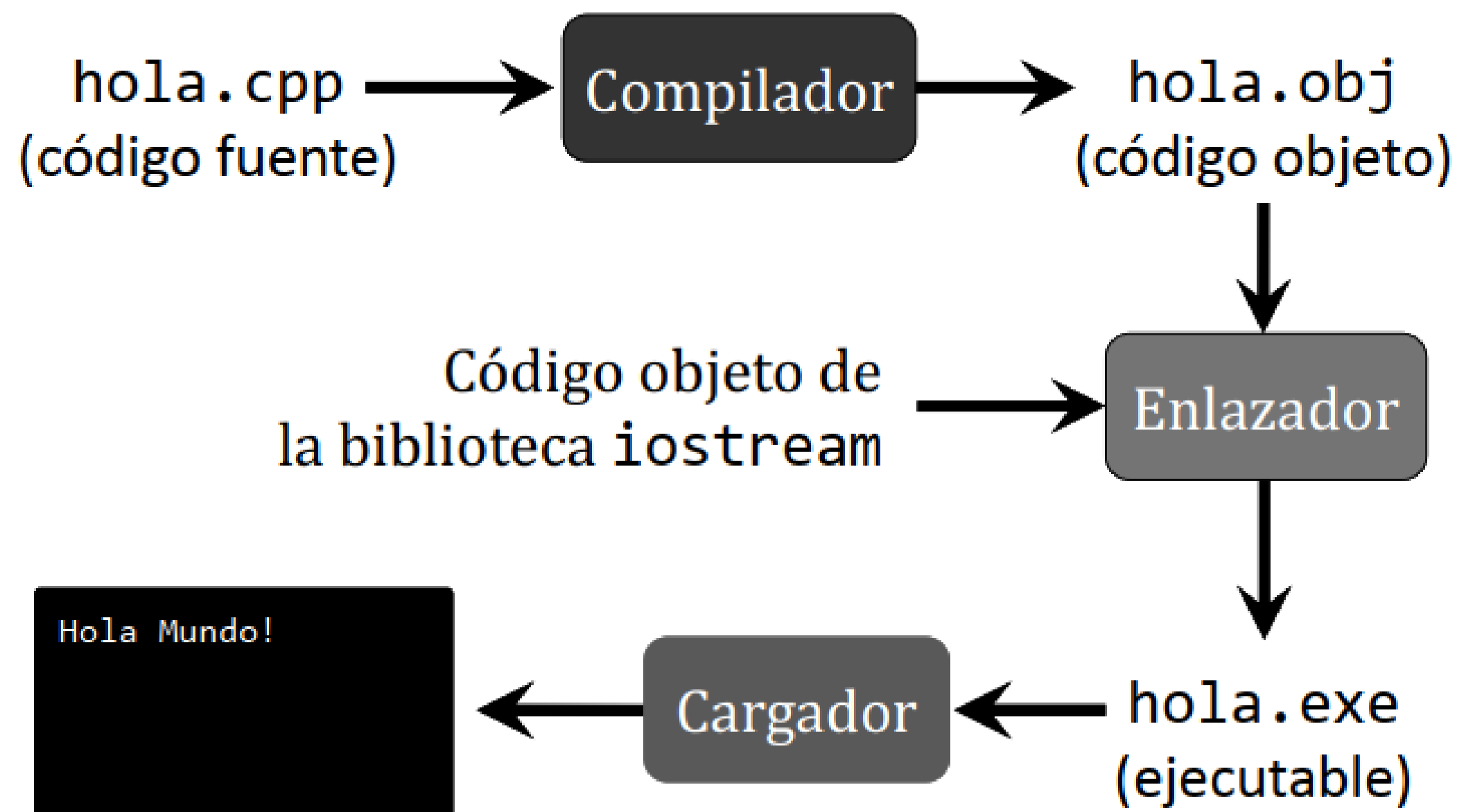
1  #include <iostream>
2  using namespace std;
3
4  int main() ...// main() es donde empieza la ejecución
5  {
6      cout << "Hola Mundo!" << endl; // Mostrar Hola Mundo!
7
8      return 0;
9  }
10

```

length: 179 lines: 10 Ln: 10 Col: 1 Sel: 0 Dos\Windows ANSI INS

Instalación y uso:
Sección
Herramientas de desarrollo
en el Campus Virtual

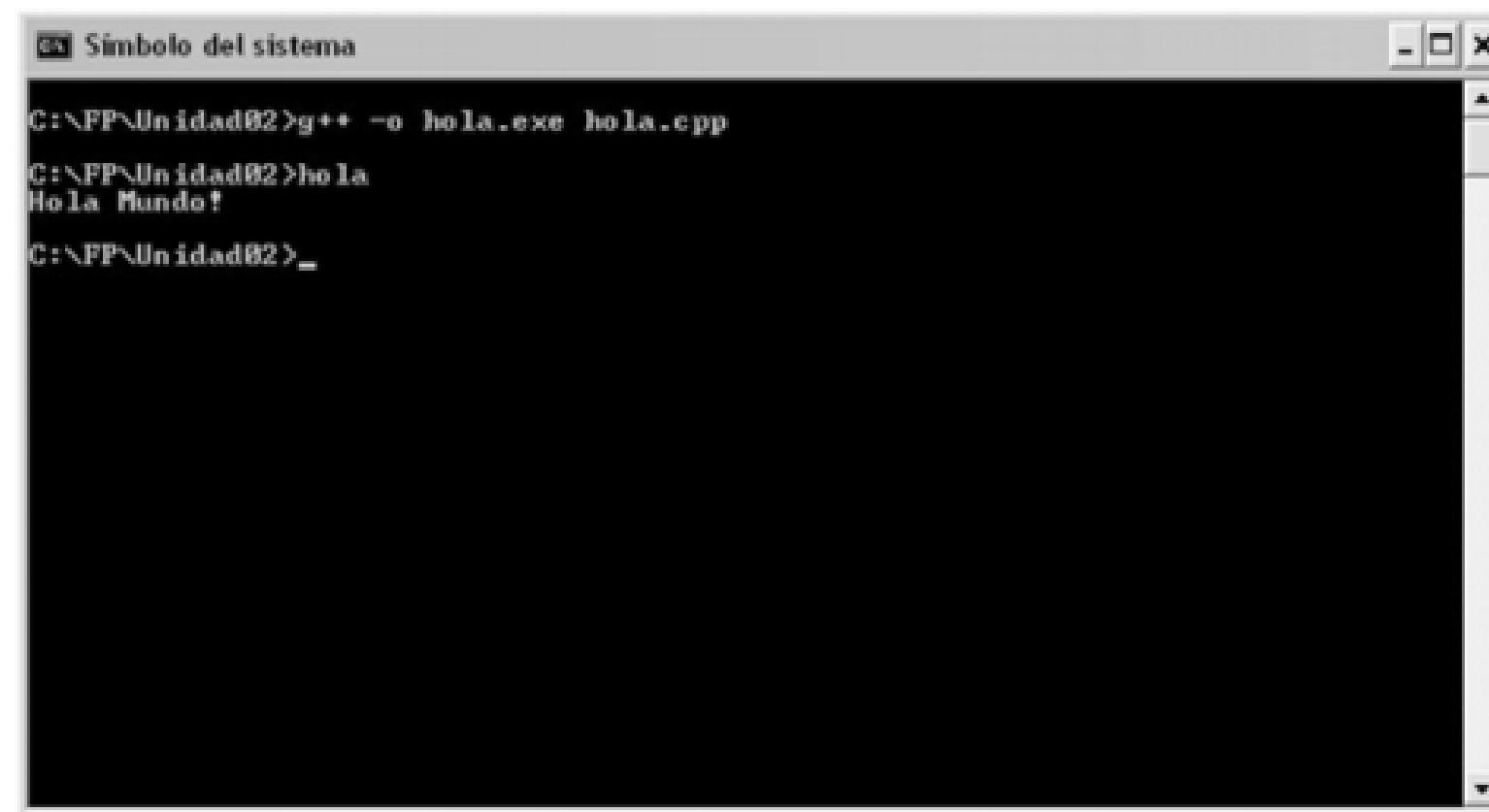
Compilación, enlace y ejecución



Más herramientas de desarrollo

Compilador

- ✓ Importante: C++ estándar
- ✓ Recomendación: GNU G++ (*MinGW* en Windows)



```

C:\FP\Unidad02>g++ -o hola.exe hola.cpp
C:\FP\Unidad02>hola
Hola Mundo!
C:\FP\Unidad02>_
  
```

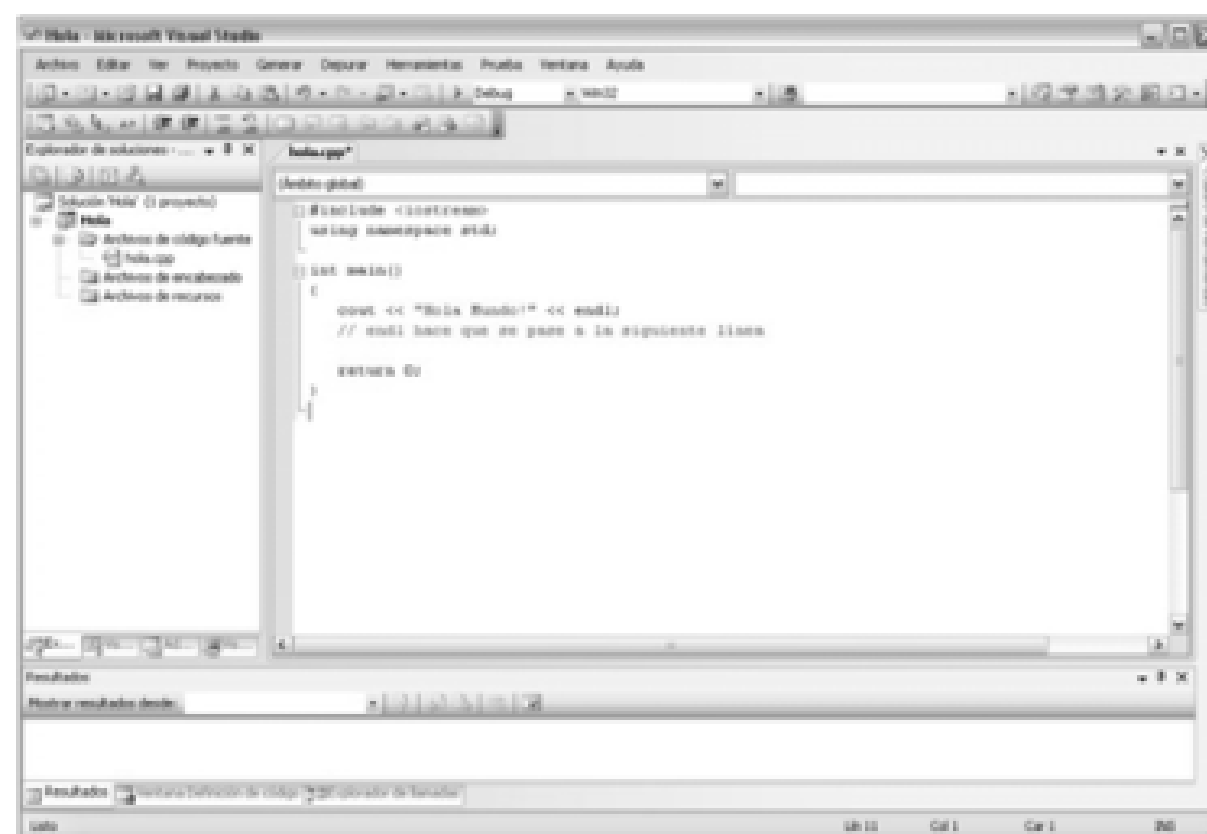
Instalación y uso:
Sección
Herramientas de desarrollo
en el Campus Virtual



Más herramientas de desarrollo

Entornos de desarrollo

- ✓ Para editar, compilar y probar el código del programa
- ✓ Recomendaciones:
 - Windows: MS Visual Studio / C++ Express o Eclipse
 - Linux: Netbeans o Eclipse



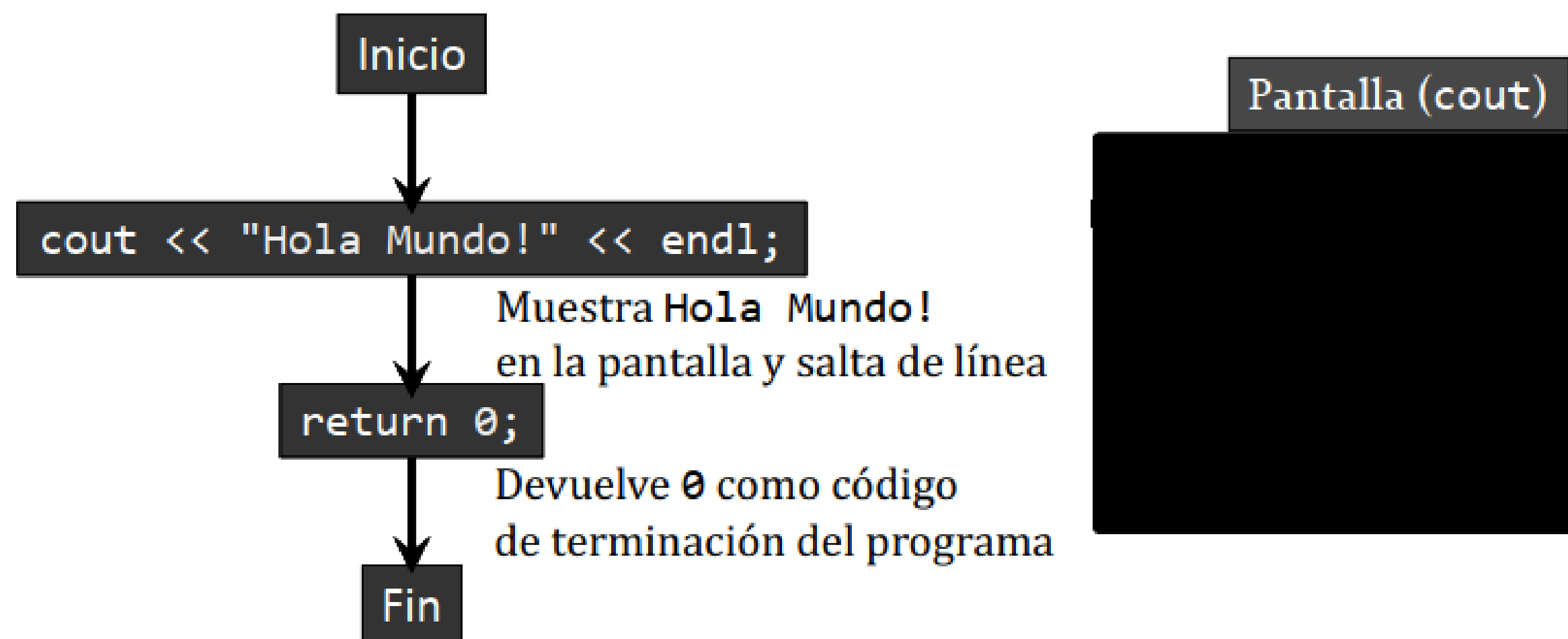
Instalación y uso:
Sección
Herramientas de desarrollo
en el Campus Virtual

Tipos e instrucciones I

Un primer programa en C++: ejecución

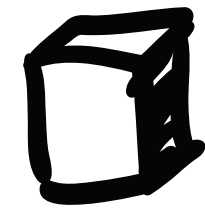
¿Qué hace el programa?

- ✓ La ejecución del programa siempre empieza en `main()`
- ✓ Se ejecutan las instrucciones en secuencia de principio a fin



Hola Mundo!

La única instrucción que produce algo



`cout (iostream)`

character output stream

Visualización en la pantalla: operador `<<` (*insertor*)

```
cout << "Hola Mundo!" << endl;
```



```
cout << "Hola Mundo!" << endl;
```

`endl` → *end line*

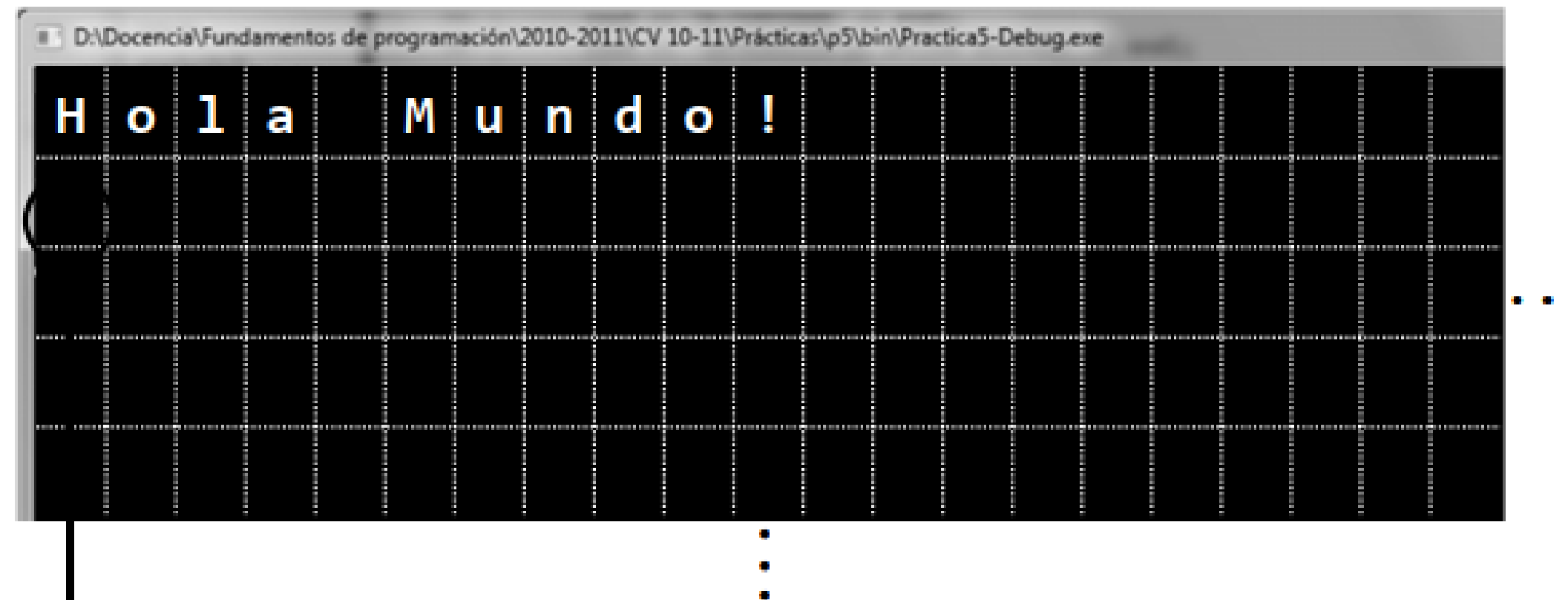
```
Hola Mundo!
```

```
—
```

Ventanas de consola o terminal

Las aplicaciones en modo texto se ejecutan dentro de ventanas:

- ✓ Windows: ventanas de consola (*Símbolo del sistema*)
- ✓ Linux: ventanas de terminal



Cursor parpadeante: Donde se colocará el siguiente carácter.

Visualización de datos

El insertor <<

```
cout << ...;
```

Inserta textos en la pantalla de modo texto

Representación textual de los datos

A partir de la posición del cursor

Line wrap (continúa en la siguiente línea si no cabe)

Se pueden encadenar:

```
cout << ... << ... << ...;
```

Recuerda: las instrucciones terminan en ;

Visualización de datos

Con el insertor << podemos mostrar...

- ✓ Cadenas de caracteres literales

Textos encerrados entre comillas dobles: "..."

```
cout << "Hola Mundo!";
```

¡Las comillas no se muestran!

- ✓ Números literales

Con o sin decimales, con signo o no: 123, -37, 3.1416, ...

```
cout << "Pi = " << 3.1416;
```

Se muestran los caracteres que representan el número

- ✓ endl

¡Punto decimal, NO coma!

El programa principal



La función `main()`: *donde comienza la ejecución...*

```
#include <iostream>
using namespace std;
```

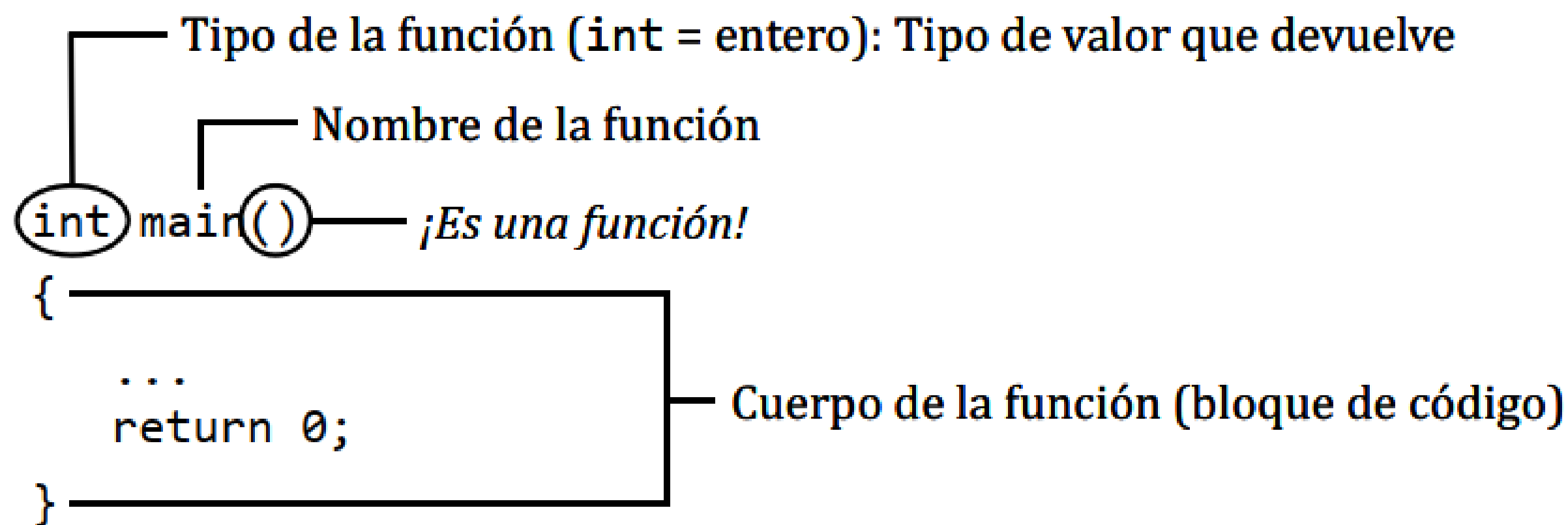
```
int main()    // main() es donde empieza la ejecución
{
    cout << "Hola Mundo!" << endl;
    return 0;
}
```



Contiene las instrucciones que hay que ejecutar

El programa principal

La función `main()`:



```
return 0;
```

Devuelve el resultado (0) de la función

Documentando el código...

Comentarios (se ignoran):

```
#include <iostream>
using namespace std;
```

```
int main()    // main() es donde empieza la ejecución
{
    cout << "Hola Mundo!" << endl;
    ...
}
```

Hasta el final de la línea: // Comentario de una línea

De varias líneas: /* Comentario de varias
 líneas seguidas */

La infraestructura

Código para reutilizar:

```
#include <iostream> ← Una directiva: empieza por #  
using namespace std;
```

```
int main()    // main() es donde empieza la ejecución  
{  
    cout << "Hola Mundo!" << endl;  
    return 0;  
}
```

Bibliotecas de funciones a nuestra disposición

Bibliotecas

Se incluyen con la *directiva* `#include`:

```
#include <iostream>
```

(Utilidades de entrada/salida por consola)

Para mostrar o leer datos hay que incluir la biblioteca `iostream`

Espacios de nombres

En `iostream` hay espacios de nombres; ¿cuál queremos?

```
#include <iostream>
```

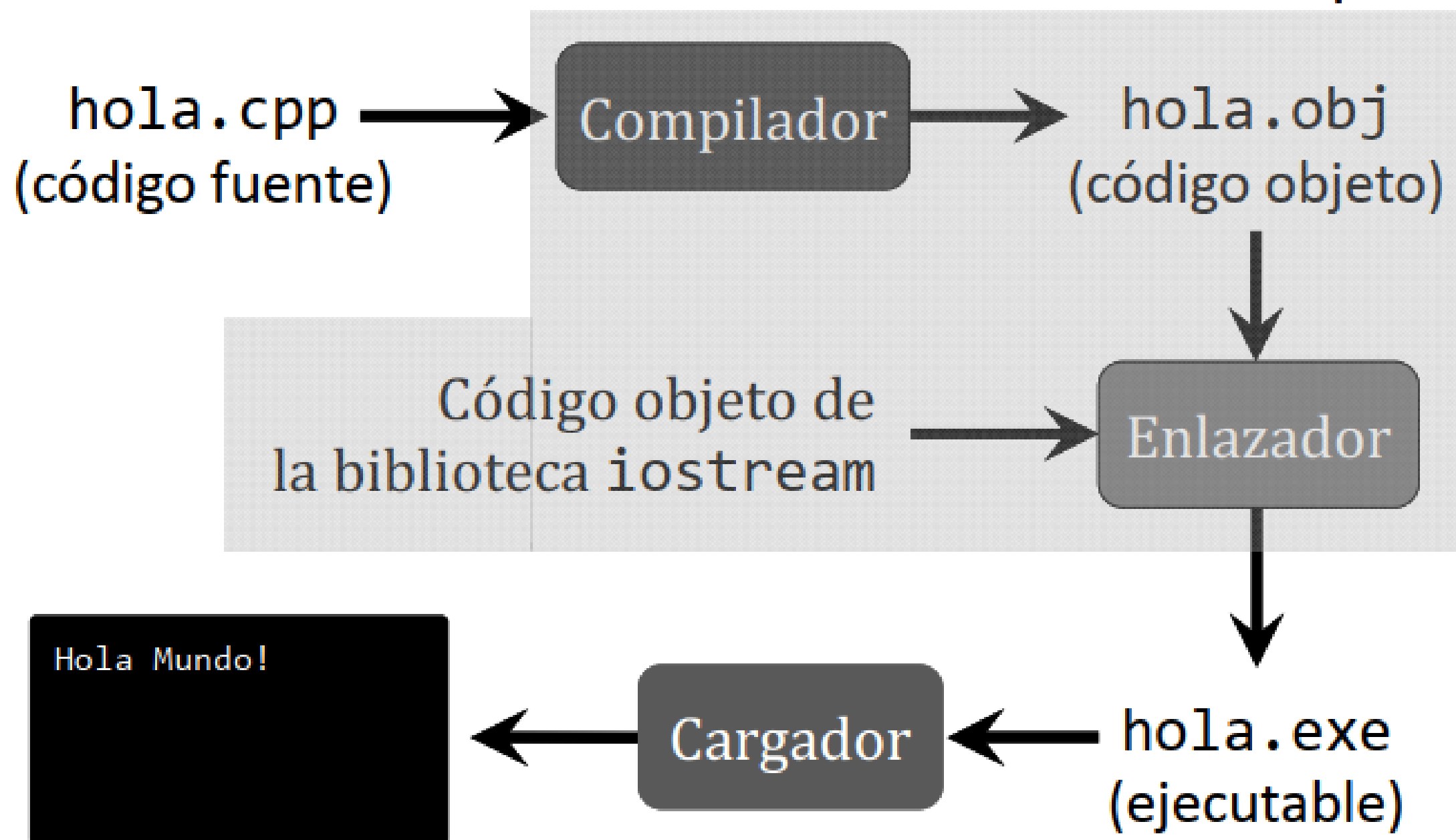
```
using namespace std; ← Es una instrucción: termina en ;
```

Siempre usaremos el espacio de nombres estándar (`std`)

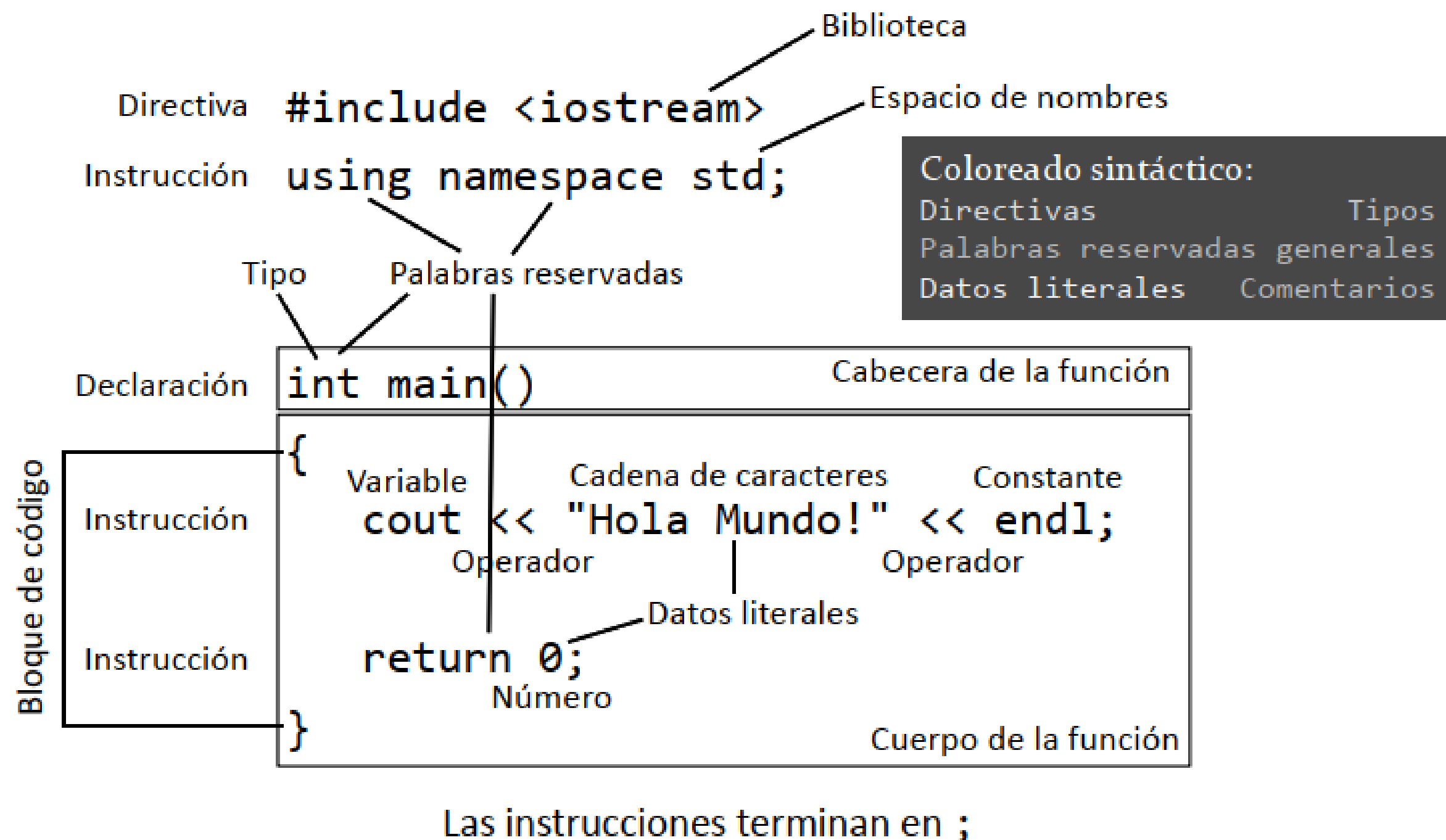
Muchas bibliotecas no tienen espacios de nombres

Compilación y enlace

A menudo en un paso



Elementos del programa



Uso de espacio en blanco

Separación de elementos por uno o más *espacios en blanco*
(espacios, tabuladores y saltos de línea)

El compilador los ignora

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hola Mundo!" << endl;
    return 0;
}
```

```
#include <iostream> using namespace std;
int main(){cout<<"Hola Mundo!"<<endl;
return 0;}
```

¿Cuál se lee mejor?

Las líneas de código del programa

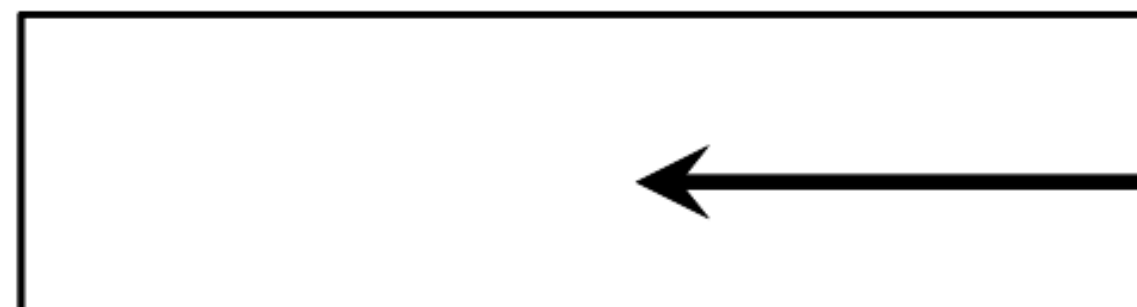
Programa mínimo

Programa con E/S por consola

Una plantilla para empezar:

```
#include <iostream>
using namespace std;
```

```
int main()
{
```



```
    return 0;
```

```
}
```

¡Tu código aquí!

El Quijote...



... recitado en la consola

Mostrar los textos con cout <<:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "En un lugar de la Mancha," << endl;
    cout << "de cuyo nombre no quiero acordarme," << endl;
    cout << "no ha mucho tiempo que vivía un hidalgo de los de
lanza en astillero, ..." << endl;
    return 0;
}
```

Líneas de código

Introducción del código del programa

Terminamos cada línea de código con un salto de línea (↵):

```
#include <iostream> ↵
using namespace std; ↵
↵
int main() ↵
{ ↵
    cout << "En un lugar de la Mancha," << endl; ↵
    cout << "de cuyo nombre no quiero acordarme," << endl; ↵
    cout << "no ha mucho tiempo que vivía un hidalgo de los de
lanza en astillero, ..." << endl; ↵
    return 0; ↵
} ↵
```

Líneas de código

Introducción del código del programa

No hay que partir una cadena literal entre dos líneas:

```
cout << "no ha mucho tiempo que vivía un hidalgo de
los de lanza en astillero, ..." << endl;
```

¡La cadena no termina (1ª línea)!

¡No se entiende los (2ª línea)!





Veamos cómo nos muestra los errores el compilador...

Programar pensando en posibles cambios

Mantenimiento y reusabilidad

- ✓ Usa espacio en blanco para separar los elementos:

```
cout << "En un lugar de la Mancha," << endl;
```

mejor que

```
cout<<"En un lugar de la Mancha,"<<endl;
```

- ✓ Usa sangría (indentación) para el código de un bloque:

```
{
Tab  → cout << "En un lugar de la Mancha," << endl;
ó
3 esp. | ...
      | return 0;
}
```

¡El estilo importa!

Cálculos en los programas

Cálculos en los programas

Operadores aritméticos

- + Suma
- Resta
- * Multiplicación
- / División

Operadores binarios

operando_izquierdo operador operando_derecho

Operación	Resultado
3 + 4	7
2.56 - 3	-0.44
143 * 2	286
45.45 / 3	15.15

Cálculos en los programas

Números literales (concretos)

- ✓ Enteros: sin parte decimal

Signo negativo (opcional) + secuencia de dígitos

3 143 -12 67321 -1234

No se usan puntos de millares

- ✓ Reales: con parte decimal

Signo negativo (opcional) + secuencia de dígitos

+ punto decimal + secuencia de dígitos

3.1416 357.0 -1.333 2345.6789 -404.1



Punto decimal (3.1416), NO coma (~~3,1416~~)

Cálculos en los programas

cálculos.cpp

Ejemplo

```
#include <iostream>
using namespace std;

int main()
{
    cout << "133 + 1234 = " << 133 + 1234 << endl;
    cout << "1234 - 111.5 = " << 1234 - 111.5 << endl;
    cout << "34 * 59 = " << 34 * 59 << endl;
    cout << "3.4 * 5.93 = " << 3.4 * 5.93 << endl;
    cout << "500 / 3 = " << 500 / 3 << endl; // Div. entera
    cout << "500.0 / 3 = " << 500.0 / 3 << endl; // Div. real

    return 0;
}
```

Un texto

Un número

Cálculos en los programas

División entera

División real

```
D:\FP\Tema02>g++ -o cálculos cálculos.cpp
Información: se resuelve std::cout al enlaza
ción)
c:/mingw/bin/../../lib/gcc/mingw32/4.5.0/../../.
importación automática se activó sin especifi
a de órdenes.
Esto debe funcionar a menos que involucre est
ferencién símbolos de DLLs auto-importadas.

D:\FP\Tema02>cálculos
133 + 1234 = 1367
1234 - 111.5 = 1122.5
34 * 59 = 2006
3.4 * 5.93 = 20.162
500 / 3 = 166
500.0 / 3 = 166.667
```

Cálculos en los programas

¿División entera o división real?

Ambos operandos enteros → División entera

Algún operando real → División real

División	Resultado
500 / 3	166
500.0 / 3	166.667
500 / 3.0	166.667
500.0 / 3.0	166.667

Comprueba siempre que el tipo de división sea el que quieres